TsunamiClaw User's Guide

David L George
Deptartment of Applied Mathematics
University of Washington

April 7, 2006

1 Introduction

TsunamiClaw is an extension of the Clawpack software, a set of fortran routines for solving hyperbolic systems of PDEs. TsunamiClaw is available for download at the Clawpack website [2]. TsunamiClaw contains many of the standard routines from Clawpack, and the adaptive mesh routines Amrclaw, as well as many specialized routines for solving the nonlinear shallow water equations, particularly for tsunami modeling. The standard TsunamiClaw library contains all of the necessary library routines, and is available at the Clawpack website. This document contains information specific to TsunamiClaw. More general documentation for Clawpack and Amrclaw is available at [2] and should be consulted before reading this document.

2 General Features of TsunamiClaw

TsunamiClaw allows one to model global-scale tsunamis and inundation on a latitude-longitude grid (or smaller local tsunamis on a Cartesian grid) with a diverse range of spatial and temporal scales. This is accomplished by using a single coarse *level* 1 grid for the entire domain, and evolving rectangular Cartesian sub-grids of higher refinement, *level* 2 , $\cdots$, *level* n, that track moving waves and inundation at the shoreline. Up to six levels may be used (though typically four or less is recommended). At any given time in the calculation, a particular level of refinement may have numerous disjoint grids associated with it. User specified integers determine the refinement ratios between particular levels, which can lead to a large degree of refinement even for a small number of levels.

The following example illustrates a typical usage of grid-level refinement for calculating the Indian Ocean Tsunami. Consider four levels with refinement ratios of 10. The entire Bay of Bengal can be covered with a *level* 1 grid with 50×50 grid cells, each $100\text{km} \times 100\text{km}$ ($1^\circ \approx 100\text{km}$). A second level might be dedicated to tracking the deep ocean tsunami with grid cells that are $(10\text{km})^2$ (this resolution is sufficient given the long wavelength of deep ocean tsunamis). Third-level grids near the shore could resolve the compressing tsunami onto $(1\text{km})^2$ grid cells, resolution sufficient to identify vulnerable coastlines. Fourth-level grids would then allow resolution onto $(100\text{m})^2$ grid cells. Alternatively, if a specific region was of interest, one could easily resolve a specific community or shoreline onto $(10\text{m})^2$ grid cells using four levels and the right refinement ratios, allowing detailed inundation studies. Calculations of this sort can easily be done on personal desktop or laptop computers using TsunamiClaw. Computing the same domain at this single highest resolution would require 10^{10} grid cells and prohibitively small time-steps, and would not be feasible given current computing technology.

TsunamiClaw is based on a finite volume numerical method, which means that the solution is represented as a piecewise constant, with numerical values approximating the average solution value in each discrete computational grid cell. There is no specific reference to the shoreline—grid cells may simply be wet or dry depending on their location, and may fill-up or drain-out of water as waves inundate or draw-down at the shoreline. This means that dry land is part of the computational domain, and a single grid is a simple rectangle that may overlap the shoreline. TsunamiClaw solves the shallow water equations in their physically relevant conservative form, therefore, the solution is represented as water depth and momentum. This is one feature that allows convergence to discontinuous bores. See [3] and [1] for a mathematical descriptions of these methods.

3 Running TsunamiClaw

This document assumes a basic working knowledge of Unix or Linux. TsunamiClaw is written in fortran 77, and can be run on any system with a proper fortran compiler, however, using the provided Makefiles requires that you are using a Unix or Linux operating system. TsunamiClaw consists of a directory of library routines, as well as an application directory that will be used to specify problem specific parameters. Routines in the library should typically not require any modification, and should only be modified at your own risk. (One possible exception is the `call.i` file. See the standard Clawpack documentation).

4 The TsunamiClaw library

The standard TsunamiClaw library,

`tsunamiclaw/lib`,

is available at the Clawpack website as a tar-zipped directory,

`tsunamiclaw.tar.gz`.

On your system, the unzipped directory should be in a path specified by an environment variable that you should name `TSUNAMICLAW`. The path to the TsunamiClaw library will therefore be referred to as,

`$TSUNAMICLAW/lib`.

It is recommended to set this environment variable in one of your login scripts (such as .cshrc or .bashrc depending on your shell). Before using TsunamiClaw, the `.f` fortran files in the library must be made into `.o` object files. This can be done by typing "`make`" from the library directory

`$TSUNAMICLAW/lib`.

(Note: the variable `TSUNAMICLAW` must be set before using `make`.) If the library is modified (not recommended), or moved from one system to another, `make` should be run again.

The TsunamiClaw library contains a mix of routines, some that are unique and some that are standard routines of Clawpack and Amrclaw. Routines that are unique to the TsunamiClaw library are named by the convention `filename_tc.f`. Other versions from the standard Clawpack library may or may not be compatible with the `filename_tc.f` routines. Use caution if updating to newer versions of the Clawpack or Amrclaw libraries.

5 Application Directory

TsunamiClaw will be run from an application directory that may reside anywhere in your directory structure. An application directory will contain input data files where parameters will be set for each specific problem. TsunamiClaw uses two input data files, `amr2ez.data` and `setprob.data`. Standard Clawpack computational parameters are set in `amr2ez.data`, such as the physical domain and refinement parameters,

and the other file, `setprob.data`, is used to set computational parameters specific to TsunamiClaw. The application directory will also contain several required fortran routines (`.f` files), explained in section section 8 that can optionally be modified to your needs, however, for standard applications it should only be necessary to set parameters in the input data file. To make the executable that runs TsunamiClaw, type "`make xtsunami`" from the application directory. The executable file `xtsunami` can then be run from this directory. An example application directory comes with TsunamiClaw,

`$TSUNAMICLAW/tc_example`,

that contains the required fortran routines (`.f` files) and the input files (`.data` files), as well as some matlab plotting scripts (`.m` files). It is recommended that you copy this directory to another location outside of `$TSUNAMICLAW` before making modifications, so that if you update TsunamiClaw to newer versions, your personal application directories will not be overwritten.

6 Data Input (`setprob.data`)

The computational parameters set in `setprob.data` are described below. Note that integer variables start with the letters `m` or `l`, so values specified in `setprob.data` for these variables should not contain any decimal points.

`gravity`: The gravitational constant should typically be set to 9.8, unless units other than meters and seconds are desired. When using the default latitude-longitude units, internal calculations rely on meters as the physical units, so the default value of 9.8 should be used.

`depthtolerance`: Water depths below this value (in meters) will be effectively treated as zero. This makes the computation more efficient, since it prevents computing the solution in areas with vanishingly small depths. It is recommended to use a reasonable value such as 10^{-3} (1 millimeter) or larger. Since the shallow water equations have no viscosity, water velocities can be unrealistically large in regions with very shallow depths, and this can lead to inefficient time-stepping or even instabilities. Using a value in the range of 10^{-3} to 10^{-2} is recommended.

`shallowdepth`: This parameter is used for refinement purposes. Typically more refinement is desired, and needed due to decreasing wavelengths, in shallow coastal waters. This parameter indicates areas that are considered "*shallow*" in meters. Larger values will lead to larger areas of refinement, while smaller values will lead to limited refinement areas closer to shore. A typical value might be 100 (meters). See `maxdeepoceanlevel` below.

`wavetolerance`: This parameter is used for refinement purposes near waves, and controls how sensitive refinement is to deviations from level sea-surface. Smaller values will therefore lead to larger areas of refinement and larger values will lead to smaller areas of refinement surrounding only the biggest waves. Since deep ocean tsunamis do not necessarily have large amplitudes, a value such as 10^{-1} (10 cm) might be a typical value for a global scale tsunami.

`frictioncoeff`: This parameter sets the Manning coefficient used for optional bottom friction. Bottom friction may be important for realistic run-up heights and can speed calculation times as well. The friction formula is a standard empirically derived formula. Higher values of the Manning coefficient correspond to greater friction. A typical value used in a tsunami calculation might be 0.025. For

friction, the source-term splitting option must be set in `amr2ez.data`. If you do not want to use friction, you can set `frictioncoeff` to 0.0, or preferably turn-off source term integration in `amr2ez.data`

`maxdeepoceanlevel:` This parameter determines what level of refinement is allowed in the deep ocean. Typically deep ocean tsunamis have very long wavelengths, and so high levels of refinement are not needed and would be computationally wasteful in the deep ocean. A typical computation might only allow the second level of refinement in the deep ocean, reserving higher level grids for shallow regions near the shore. For classification of deep and shallow TsunamiClaw uses the `shallowdepth` parameter.

`levelallowdomain:` This parameter determines what level of refinement is allowed on the entire domain. For instance, it might be desired to restrict high levels of refinement to certain regions of the domain. In this case tt levelallowdomain should be set to some lower value that is allowed on the entire domain. See below for parameters that allow higher refinement in certain regions.

`methodquake:` This parameter determines the type of method that is to be used to generate the tsunami. It can have integer values of 0,1, or 2. If `methodquake`= 2, then a dynamic fault model is used, as specified in a user provided data file. If `methodquake` = 1, then a static initial condition is used, as specified in a user provided data file. The formats for these files are described in the section "Fault Models". If `methodquake`= 0 then the user must provide some other means of generating the tsunami, such as specifying an initial condition directly in `qinit.f`, or perhaps a user specified boundary condition. To use these latter options, see the standard Clawpack documentation.

`minfaultrefinelevel:` This parameter is used for determining refinement in the region described by the fault model. It can have any integer value up to the number of refinement levels specified in `amr2ez.data`. Setting `mfaultrefinelevel`= n will enforce that level n grids or finer will cover the fault region during the fault motion. If you do not want to enforce a particular level, but wish for TsunamiClaw to treat the fault region like any other region, refining based on wave heights and water depths, then set `mfaultrefinelevel` to a low value such as 0 or 1.

`faultfname:` On the line following `mfaultrefinelevel`, you should place the relative or full path to the file of the fault model in single quotes. If `methodquake`= 0, then the line should be blank.

`mbathfiles:` Set `mbathfiles` to the integer number of bathymetry files that you would like to read in. If you are specifying bathymetry directly by some other means, set `mathfiles` = 0. See the description of `setaux.f`.

`levelallowbathfile(mbathfiles):` This line should contain a list of `mbathfiles` integers, specifying the level of refinement allowed over each bathymetry file to be used. This variable is useful if you would like to use a certain level of refinement over most of the domain, yet reserve high levels of refinement for regions where you have more detailed bathymetry. For more about how bathymetry files can be used, see the section Bathymetry.

`bathfname(mbathfiles):` The next `mbathfiles` lines should contain the paths to the bathymetry files in

single quotes. There should be at least one, unless `mbathfiles` = 0, in which case the line should be empty. The order that they are listed in should correspond to values specified in the list of integers.

`mregions`: For additional control and restriction of refinement to particular regions, set `mregions` to an integer number of regions that you would like to specify.

`levelallowregion(mregions)`: This line should contain a list of `mregions` integers, specifying the level of refinement allowed over each region to be specified. This option is useful if you would like use high levels of refinement for regions of interest, even if these regions do not correspond specifically to bathymetry files.

`regioncoords(4,mregions)`: The next `mregions` lines should each contain a list of four coordinates specifying the particular region: `xlower xupper ylower yupper`.

7 Data Input (`amr2ez.data`)

The file `amr2ez.data` is used to set standard computational parameters for CLAWPACK and AMRCLAW. Consult the standard CLAWPACK documentation, particularly the section on AMRCLAW, for complete information about this file. In this section, a few of these parameters will be described that should have specific values for TSUNAMICLAW.

`method(5)`: This variable specifies whether to integrate a source-term. For TSUNAMICLAW, setting this to 1, will incorporate friction. Otherwise set this variable to 0. Using friction will improve stability, and is recommended when trying to speed the calculation or use anisotropic time refinement explained in section 12.

`meqn`: This indicates the number of equations to be solved. For TSUNAMICLAW this should be 3, since the shallow water equations are solved.

`mwaves`: This indicates the number of waves in the solution. For TSUNAMICLAW this should be 3 also.

`mcapa`: This should be set to 2, when using a latitude-longitude grid, or 0 otherwise. See section 9 for more information.

`maux`: This should be set to 3 (or greater if you add extra auxiliary variables), when using a latitude-longitude grid, or 0 otherwise. See section 9 for more information. When solving on a latitude-longitude grid, `maux` should be followed by three lines with words in quotes: "center," "capacity," and "yleft" respectively. Consult the standard CLAWPACK documentation for more information about auxiliary variables.

`t0`: This specifies the initial starting time. It is recommended to use 0, or perhaps a positive number. If you are performing a restart from an old run, then reset this value to the new starting time when using a dynamic fault model. See section 11 for an explanation, and see the standard CLAWPACK for information about how to perform a restart.

8 User Accessible Fortran Routines

This section gives a brief description of the fortran routines (`.f` files) in the application directory. These files can optionally be modified for your particular application if the default behavior of TsunamiClaw is not sufficient.

`setprob.f` This routine performs all of the data input at the beginning of the computation from `setprob.data`. It should typically not need to be modified, but could be used to input additional data.

`qinit.f` This routine initializes the solution at the start of the computation (except in the case of a restart). If you use `methodquake` $= 1, 2$ then you will not need to modify this routine. If you wish to initialize the solution in some other manner, then this is the relevant routine to use. More information can be found in the standard Clawpack documentation.

`setaux.f` This routine initializes auxiliary arrays that can be used to define some other quantity in computational grid cells. TsunamiClaw uses `aux(i,j,1)`, `aux(i,j,2)` and `aux(i,j,3)` for internal purposes, such as bathymetry and capacity functions associated with a latitude-longitude grid. If you would like to use additional auxiliary arrays for some other quantity see the standard Clawpack documentation.

`b4step2.f` This routine is called before each new time step on each grid. If you need to perform a check on the solution or output information, this is a good place to do it. For instance, this is a convenient place to make calls to user-written routines, or to output the solution if the standard output is not sufficient.

`allowflag.f` This routine is used for adaptive refinement purposes. Refinement parameters, such as the number of refinement levels and refinement ratios are specified in `amr2ez.data`. Amrclaw refines by flagging cells on a particular grid at a particular level that meet certain criteria (such as wave-height in TsunamiClaw). A clustering algorithm then groups flagged cells into rectangular regions which will then be refined to the next higher level. `allowflag.f` controls whether a cell at location (x, y) and time t is allowed to be flagged. In the default version of TsunamiClaw, `allowflag.f` is given information (described in section 6) from `setprob.data` to determine which areas are allowed to be refined. If the standard options are not sufficient, `allowflag.f` can be modified to meet special needs.

`flag2refine.f` This is the routine that actually flags the cells if they are allowed to be flagged by `allowflag.f`. The standard `flag2refine.f` in TsunamiClaw uses parameters form `setprob.data`. Modify this routine if you would like to refine based on some other criteria.

9 The Computational Domain

TsunamiClaw can compute on either a latitude-longitude grid or a Cartesian grid. In the former case, the latitude-longitude computational units are mapped internally to spherical physical space using a capacity function that takes into account the true physical geometry of each computational grid cell. Use of this capacity function requires that `mcapa` $= 2$ in `amr2ez.data`, and `maux` ≥ 3. When computing on a latitude-longitude domain the bounds of the domain are specified by longitude (x), and latitude (y) in `amr2ez.data`. Latitude should be specified in the standard -90° to 90° convention, where the southern latitudes are

negative. Although it is theoretically possible to compute all the way to the poles, this code has not been tested at extreme latitudes (because of vanishing grid cells at the poles, some numerical issues may arise if computing at extreme latitudes). Typically it is possible to use either the $-180°$ to $180°$ convention for longitude, or the $0°$ to $360°$ convention, where in both cases, the prime meridian corresponds to $0°$ and longitude increases toward the east. If the computational domain crosses the dateline however, and does not wrap around the entire globe, then the $0°$ to $360°$ convention must be used, so that the longitude of the computational domain increases logically. If the computational domain wraps around the entire globe in longitude, then periodic boundary conditions in x should be specified in `amr2ez.data`. In this case, $-180°$to $180°$, degree units could be used for longitude as well, with the computational domain being $-180°$to $180°$, however, it is preferable to place the computational boundary at $0°$ at the prime meridian rather than at $180°$ in the middle of the Pacific Ocean which is much more likely to be of interest. When computing on a latitude-longitude grid, output units are in meters and seconds. This cannot be adjusted because of internal calculations.

If you wish to compute on a Cartesian domain, then the physical space and computational space are the same, and domain bounds are set in meters or some other desired unit. It is important that `mcapa` $= 0$ in this case. When computing on a Cartesian domain, units are selected by adjusting the gravitational constant in `setprob.data` appropriately. (See the standard CLAWPACK documentation for more details about the `mcapa` and `naux` variables).

10 Bathymetry

The user must provide a file or files that contain the bathymetric and topographic data. The names of these numeric ascii text files are specified in `setprob.data`. When computing on a latitude-longitude domain, the files should contain three columns: longitude, latitude and elevation (in meters above sea level so that seafloor values are negative). The files must conform to the standard GIS format, where data begins in the northwest corner and advances in longitude first toward the northeast corner before moving to the next latitude south. This code assumes that there are no missing values, either above or below sea level, and that data points are uniformly spaced on a rectangular domain (in latitude-longitude space). If your bathymetry data does not conform to this standard, you must preprocess your data appropriately by interpolation of some sort, before using the files in TSUNAMICLAW.

When computing on a Cartesian domain, bathymetry should be supplied in the same logical format; 3 columns: x, y and elevation, advancing in the x direction first, from the largest y value to the smallest. Units should be compatible with the gravitational constant.

At the start of the computation, TSUNAMICLAW will read in all of the bathymetry files indicated in `setprob.data` and put the data into arrays. Separate bathymetry files can be nested or overlap in any manner desired, as long as the union of all of the files is at least as large as the computational domain specified in `amr2ez.data`. Typically there will be at least one file that has bathymetry covering the entire computational region, this isn't strictly necessary however. TSUNAMICLAW will determine the bathymetry in computational cells by interpolating or averaging the bathymetry arrays, depending on the relative resolution of the bathymetry and the computational cell. When two or more bathymetry arrays overlap the same area, TSUNAMICLAW will use the array with the highest resolution.

It is possible to specify idealized bathymetry (such as a simple function) rather than using a data file. (See `mbathfiles` in `setprob.data`)

11 Fault Models

TSUNAMICLAW is capable of using dynamic fault models to initialize the tsunami. The dynamic model

should describe the time-dependent seafloor displacement in some region. If a dynamic model of this form is used, set `methodquake` = 2 in `setprob.data`. This model should be specified in a numeric ascii text file with four columns: time, longitude, latitude, and displacement (from undisturbed bathymetry in meters). The time column should advance the most slowly, and longitude and latitude should advance in the standard GIS way as described in the Bathymetry section. The spatial domain of the fault model (the last 3 columns at each given time) should conform to the same standards as bathymetry described in the previous section. That is, all values on a rectangular domain should be specified, and should be uniformly spaced in latitude and longitude. The time values in the first column, should be uniformly spaced as well. If your fault model does not conform to these standards, then you should preprocess you data before using TsunamiClaw.

It is also possible to use a static initialization. In this case the bathymetry will not be altered dynamically, and the tsunami will be initiated by an initial condition that describes the sea surface at the start of the computation. This model should be specified in a numeric ascii text file with three columns corresponding to longitude, latitude and displacement of the sea surface (in meters). The longitude and latitude columns should advance in the standard GIS way, and should conform to the same standards described above. If this option is used set `methodquake` = 1 in `setprob.data`.

It is possible to instantaneously displace the sea-floor at some time after the start of the computation, in order to generate the tsunami. If this option is desired, use the dynamic fault model option. The dynamic fault file would then just have a single time in the first column corresponding to the time of displacement.

If you would like to generate the tsunami in some other way, modify `qinit.f` as needed, or enforce an appropriate boundary condition. Boundary conditions can be easily and stably enforced in the Clawpack software packages by simply specifying ghost cell values that correspond to the edge of the domain. See the standard Clawpack documentation for more details.

A NOTE ABOUT RESTARTS: Clawpack has a restart feature that allows a computation to be resumed from a previous computation. This is very useful in TsunamiClaw since, often, a single real tsunami may be computed many times for studying different regions. When a restart is used in conjunction with a dynamic fault model, it is important to change the initial time in `amr2ez.data` to the computational time at the beginning of the restart. This is so that the state of the bathymetry at the end of the previous computation can be resumed correctly.

12 Time-Stepping and Anisotropic Refinement

The number of grid levels and refinement ratios between grid levels are set in `amr2ez.data`. The variable `mxnest` sets the maximum number of grid levels, and the variable `inrat` is a list of (`mxnest` -1) integers specifying the refinement ratios between these levels. The default behavior of amrclaw is to refine isotropically in space and time. For general hyperbolic conservation laws, maintaining stability (and accuracy) typically requires this equal refinement in time. That is, a grid at *level* n must typically take `inrat(n-1)` time-steps for each time-step of *level* $n-1$ grids. This restriction is due to the CFL condition which, in the case of hyperbolic problems, depends on the speed at which characteristic information propagates. For the shallow water equations, characteristic wave speeds are related to the depth of the water. In *shallow* water, wave speeds are slower, which means that grids in water that is much more shallow than the rest of the domain could take much larger time-steps while still maintaining stability and accuracy. Therefore, if grids at a particular level are restricted to shoreline regions, or regions with relatively shallow water compared to grids at the next lower level, those grids would not typically require refinement in time as large as refinement in space. For this reason, it is possible to select anisotropic refinement in TsunamiClaw. To select this option, `mxnest` should be preceded by a negative sign in `amr2ez.data`. In this case, three lists of integers, `inratx`, `inraty` and `inratt` are required in `amr2ez.data`. The ratios for `inratx` and `inraty` would typically be identical, but `inratt` may have some integers that are smaller for levels that are restricted to shallow

regions. Note that the variables `maxleveldeepocean` and `shallowdepth` in `setprob.data` allow control of grid level restrictions to shallow regions. By carefully tuning `shallowdepth`, `maxleveldeepocean`, `inratx`, `inraty` and `inratt`, one can greatly speed calculations and still maintain stability. An example is shown below.

Suppose that the following values are set in `setprob.data`:

```
⋮                    ⋮
1.0d2                shallowdepth
⋮                    ⋮
2                    maxleveldeepocean
⋮                    ⋮
```

Then it might be feasible to reduce time-steps taken on *level* 3 grids since they are restricted to shallow regions, by setting the following variables in `amr2ez.data`:

```
⋮                    ⋮
-3                   mxnest
6 6                  inratx
6 6                  inraty
6 2                  inratt
⋮                    ⋮
```

In this case, *level* 3 grids will be refined spatially in x and y by a factor of 6, but will only take 2 time-steps for every time-step taken on *level* 2 grids. Since the ocean may have depths of up to 4000 meters, it might be possible to get away with even fewer time-steps, possibly 1. (Since wave speeds are related to $\sqrt{depth}$, a rough calculation suggests that wave speeds will be faster by a factor of $\sqrt{4000/100} \approx 6$ in the deep ocean, implying that time-steps on *level* 3 might approach that of time-steps on *level* 2. However, wave speeds are also related to water velocity, which can be quite large near the shoreline, so generally one should be more conservative. When reducing time-step refinement, careful attention should be paid to the Courant numbers reported for each level (which can be output by selecting the `verbose` options in `amr2ez.data`) .)

13 Memory Allocation (`bathcommon.i`)

Memory allocation for bathymetry and other arrays in TsunamiClaw is specified in an include file that resides in the application directory, named `bathcommon.i`. If the bathymetry is larger than the space allocated in `bathcommon.i`, then the calculation will abort and you will be instructed to increase memory. You must recompile the executable afterward, even if make doesn't recognize that the code has been modified. At the beginning of each computation, TsunamiClaw will output some information to the standard output about how much memory is allocated for bathymetry, and how much memory is actually used. If more memory is allocated than needed, you can abort the calculation and modify `bathcommon.i` if you wish to allocate less memory.

14 Data Output

TsunamiClaw uses the same output format as the standard Clawpack software. At user specified times (set in `amr2ez.data`) the entire solution on every grid is output into an ascii text file `fort.qnnnn`, where the `nnnn` ranges from 0000 for the initial time to the number of output times. Information about this file, such as the time and the number of grids, is output into `fort.tnnnn` with a corresponding number. These files conform to a standard format that can be read by Matlab graphics routines for visualization (see section 15). The solution in `fort.qnnnn` is grouped by grids—the solution on a particular grid is written in several columns following some header information about that grid's size and location. Each column of the solution data represents a variable, and each row represents a particular grid cell. The solution data for a grid starts at the lower left corner and advances in the x-direction first (indexed by `i`). In TsunamiClaw, the solution is output as four variables in four columns: `q(i,j,1)` is the water depth, `q(i,j,2)` and `q(i,j,3)` are the momenta (depth times water velocity) in the x and y directions respectively, and `q(i,j,4)` is the surface elevation, where 0 is mean sea level. Note that having the surface elevation and the water depth allows determination of the bathymetry in a grid cell, since depth subtracted from surface elevation gives the elevation of the bathymetry or topography. Also, since TsunamiClaw solves the shallow water equations in their physically relevant conservative form in order to allow convergence to bores, momenta are output rather than the more common variable of water velocity. Velocity can be recovered by dividing the momentum by depth. Some care should be exercised when doing this, since velocities at vanishing depths may not be accurate, and division by zero for dry cells should obviously be avoided.

When using the latitude-longitude option to specify the computational domain, all output quantities are in units of meters and seconds. Therefore, in a particular grid cell, the momenta output in the solution are in the locally orthogonal longitude and latitude directions, and given in meters squared per second.

15 Matlab Graphics and Output Visualization

TsunamiClaw comes with a set of Matlab plotting routines, `.m` files, in the directory

`$TSUNAMICLAW/matlab`,

as well as a few routines in the example application directory. These routines are a mix of the standard Clawpack matlab plotting routines as well as several specialized routines for plotting with TsunamiClaw. If you use Clawpack software for other applications, then the `$TSUNAMICLAW/matlab` directory must be higher on your Matlab path than the standard Clawpack Matlab routines, since some modified routines share the same name. (Much like Unix itself, which uses the `PATH` variable to specify its directory search path, Matlab uses the variable, `MATLABPATH` when run under Unix, which can also be set in your login scripts.)

TsunamiClaw uses the Clawpack Matlab script `plotclaw2.m` as the main plotting routine. Run `plotclaw2` from the application directory (the file for the routine resides in the TsunamiClaw Matlab library). A number of plotting options are specified in `setplot2.m`, which resides in the application directory (see Clawpack documentation for descriptions and examples). `plotclaw2` calls `setplot2` if you select `yes` when queried from the Matlab prompt. For some specialized TsunamiClaw plotting features, set `PlotType=5` in `setplot2.m`. This creates a colored surface plot of the water surface and land, using some colormaps provided in $TsunamiClaw/matlab. Waves are shaded from red at the peaks to dark blue at the troughs. The default view is directly overhead, but note that these surface plots can be rotated and zoomed, either manually or by adjusting Matlab's `view` properties. Specific user issued plotting features, such as axis labeling, axis limits, color axis properties for the wave shading, or other options, can be specified in `afterframe.m`. The standard Clawpack documentation provides a full explanation of how to use the Matlab plotting features.

16 Acknowledgements

TsunamiClaw is an extension of the existing Clawpack software, including Amrclaw adaptive mesh refinement routines and Matlab plotting routines. The two-dimensional Clawpack routines were written by

Randall J. LeVeque
University of Washington.

The adaptive mesh refinement algorithms were written by

Marsha Berger
Courant Institute, NYU.

Many of the Matlab plotting routines were written by

Donna Calhoun
CEA, Saclay France.

Numerous other people have contributed to Clawpack and Amrclaw in a variety of ways. Development of this software has been supported by grants from the NSF and the DOE.

17 Usage Restrictions

References

[1] M. J. Berger and R. J. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Numer. Anal.*, 35:2298–2316, 1998.

[2] R. J. LeVeque. Clawpack software. `http://www.amath.washington.edu/~claw`.

[3] R. J. LeVeque. *Finite Volume Methods For Hyperbolic Problems*. Cambridge University Press, 2002.